RETDAT-GETS32 Data Options

Choices for user

Robert Goodwin

Tue, Mar 18, 2008

After reviewing the NServer code in the ACReq task that supports Acnet data request protocols, as described in the note, *NServer Logic Flow*, it was apparent that many options must be handled by that code. This note takes the opposite approach of itemizing the choices for access to data that are available.

Acnet properties

There are only a few properties defined for access to data in Acnet. The main property is the reading property, for which an 8-byte SSDN identifies for the front end what data is sought. Each named Acnet device has only a single reading property and therefore one SSDN. When a data request is made from an Acnet client, the 8-byte SSDN, obtained from the Acnet database, is passed to the front end from the client application; the application cannot modify this structure. The only options available to the application are the length, offset, and FTD, or frequency time descriptor. The front ends can use only these three parameters to determine what type of data is to be returned for a given device reading property.

Various options

Here is a list of available options for accessing data, along with how it can be done. Assume that the SSDN has the form

```
00o1 node chan 00sz
```

The hi byte of the first word is the listype# that the front end understands. In these cases, it has the value 0x00, which corresponds to analog channel readings. Besides the node and channel that specify the reading of interest, one has an optional offset parameter and an optional array parameter. The offset parameter o can have the value 0, 1, or 2. The array parameter sz specifies the size of each array element and, if nonzero, is usually 0x02.

Offset option:

If offset option o = 1, the offset value is added to chan. If o = 0, the option is not used in *this* way, although it may be used as a byte offset into a waveform array. See below.

Array option:

If array option sz = 0x02, that the request seeks an array of (length/2) 16-bit readings, sampled from consecutive channels, beginning with chan. The value of 0x02 refers to copying 2 bytes from each channel entry in the ADATA table, starting at the reading field. When sz is 0x00, the array option is not available.

Ordinary 2-byte reading

length = 2, offset = 0, period = any

If periodic slower than 7.5 Hz (2 cycles), beam-preferred averaging applies, which means that for each 15 Hz cycle over the indicated period, an average of beam cycle data is built. In the case that no beam cycle occurs within the period, an average of all (non-beam) cycles is built. A beam cycle is indicated by Bit 0x9F, where 0 means beam, 1 = non-beam. (In a sense, then, it might be considered “non-beam status.”)

Waveform data

Note that for access to any waveform, there must be an entry in the CINFO table that associates the indicated channel number with a waveform, so the code can locate it. These cases are for Swift/Quick/Quicker digitizers, in which one has an array data of points, and the sample rate is known implicitely. (For the KHz digitizers, 1KHz or 10KHz, access methods are described later.)

length > 2 , offset option = 0, offset = any, period = any, but not 7.5 Hz

In this case, the offset is interpreted as a byte offset from the start of the array of 2-byte point values. The number of data points returned is length/2.

length = 2, offset > 0

The returned data is a single point of a waveform, except the first one. (Note that offset = 0 would be interpreted instead as an ordinary 2-byte reading from the data pool.)

Time-stamped data

length = 8, offset = 0, period = 7.5 Hz

Structure returned is 4 words: the #readings (1 or 2), the cycle# of the first reading, and one or two readings sampled from consecutive cycles. Note that this scheme returns two cycles of readings every other cycle, a reliable means of delivering 15 Hz correlated data. (The reason for the #readings word in the reply is that the first-time reply is immediate and therefore includes only a single reading.) A useful example is a reading of B:BREVNT, which returns the current Booster reset event#.

length >= 8, offset > 0, period = 7.5 Hz

This can return a waveform segment captured on two consecutive cycles. The number of segments (1 or 2) is followed by the cycle number of the first segment, which is followed by two successive cycles of waveform segment data. The number of points returned in each segment is (length–4)/4. To read 400-point waveforms in this way, use length = 1604.

Cycle data

length >= 2, offset = 0, period = any, but not 7.5 Hz (unless length = 6)

This returns an array of 2-byte readings, where the first word is the current cycle reading, the following words are readings of previous cycles, and the last word is the cycle number of the current cycle. For example, a request for 8 bytes will return three readings in reverse cycle order, starting with that of the current cycle, followed by the current cycle number. If one asked for such replies every 3 cycles (5 Hz), then one could capture all 15 Hz data, and the cycle number would allow for data correlation. Only the last 32 cycles of 2-byte data is maintained as Cycle data, so the maximum value of length is 66.

Note that for a length of 6 bytes, one can use a period of 7.5 Hz, in order to receive Cycle data every other cycle, but if one asks for more than 6 bytes at 7.5 Hz, one gets instead the reply data formatted as above under "time-stamped data."

Event-based requests

For an event-based request, the data is returned on the cycle following the occurrence of the given event. In many front ends, the events of interest occur at Booster reset time, and

the front end "μP-Start" time is set for a delay of milliseconds, usually in the range 3–40. The data returned is that found in the data pool on the current cycle. For event-plus-delay requests, available with the GETS32 protocol, the cycle on which the data is sampled may be delayed one or more cycles, according to the value of delay, which is in milliseconds.

Note that for testing purposes, there is a limited ability to ask for event+delay using the RETDAT protocol. A 7-bit delay in milliseconds can be specified in bits 14–8 of the FTD.

KHz digitizer buffer event-based requests

Special support is included for data sampling from the IRM 1KHz hardware buffer or from the HRM 10 KHz buffer. This support is only granted requests that are based upon clock event (or state variable) plus delay, which implies the GETS32 protocol, except that the informal RETDAT extension can also be used for testing. If the delay is zero, then the reading is taken from the data pool on the cycle for which the event is true, just as for the RETDAT protocol; otherwise, it is sampled from the hardware buffer according to the given event plus delay. Note that for this support, the array option cannot be used; *i.e.*, `sz = 0x00`.

Floating point cases

For the following, the SSDN has the following format:

```
5Ao1 node chan 00mm
```

The listype #90 indicates raw floating point channel readings. Such 4-byte data is calculated by a local application and deposited into the FDATA table instead of the ADATA table. There are some similar options available for access to this data as for integer data.

Offset option:

If offset option `o = 1`, the offset value is added to `chan`.

Array option:

If array option `sz = 0x04`, it means that the request seeks an array of (length/4) 32-bit readings, sampled from consecutive channels, beginning with `chan`. The value of `0x04` refers to copying 4 bytes from each channel entry in the FDATA table, starting at the reading field.

Ordinary 4-byte readings

length = 4, offset = 0, period = any

This returns raw floating point readings sampled from the FDATA table on this cycle.

Time-stamped data

length = 12, offset = 0, period = 7.5 Hz

Structure returned is 6 words: the #readings (1 or 2), the cycle# of the first reading, and one or two 4-byte readings sampled from consecutive cycles. Note that this scheme returns two cycles of raw floating point readings every other cycle, a reliable means of delivering 15 Hz correlated data.

Cycle data

length >= 4, offset = 0, period = any, but not 7.5 Hz

This returns an array of 4-byte readings, where the first word is from the current cycle reading, the following words are reading of previous cycles, and the last word is the (floating

point) cycle number of the current cycle. For example, a request for 16 bytes will return three readings in reverse cycle order, starting with that of the current cycle, followed by the current cycle number. If one asked for such replies every 3 cycles (5 Hz), then one could capture all 15 Hz data, with the cycle number allowing for data correlation. Only the last 16 cycles of 4-byte Cycle data is maintained for raw floating point channels. The maximum value of length is therefore 68.

KHz digitizer waveforms

For this case, the `FTPMAN` snapshot protocol is usually used, although it cannot be used at fast rates. But a carefully crafted `RETDAT` request can work. The `SSDN` has to look like this:

```
52o2 node chan 00vv
```

The channel number has to be marked in a `CINFO` table entry as being associated with one of these digitizers. (This uses listype #82.) The format of the reply data structure is as follows:

Field	*Size*	*Meaning*
`tBase`	4	32-bit time stamp in 10-μs units
`tReset`	4	32-bit time stamp if new event occurs within following data
`data`	2	first data point
`time`	2	first time stamp, offset from tBase also in 10 μs units
`data`	2	second data point
`time`	2	second time stamp
etc		

The length is chosen based upon the desired sample rate and the reply period. For example, to read 1 KHz data from the 1 KHz digitizer, with data to be returned at 15 Hz, the length should be (8 + 4*66) = 272.

The clock event is specified in the 4th word of the `SSDN`. One can specify offset option `o = 1` to allow using the offset parameter to specify the clock event, if `vv = 0x00`. If offset option `o = 0`, then `vv` specifies the fixed clock event#.

As an update to the request is made, data points are sampled from the 1KHz buffer, say, at the implied sample rate that is given by the length and the reply period. The sampling that is done brings the sampling up-to-date. It is possible that points can be skipped or duplicated in order to accomplish this, but the time stamp associated with each point is always correct. Note that the first reply for a periodic case is delivered promptly, such that almost no digitization has taken place since the request was intialized. Therefore, in that first reply, nearly all points in the buffer may be duplicates, with corresponding duplicate times.

The reason for the second long word in the reply buffer (`tReset` above) is to mark a time when the reset event occurred during sampling of the data points to fill the buffer. The full time stamp is always given as `tBase+time` for each point. If `tReset` is nonzero, the client should notice, when processing the data points, that when `tBase+time` reaches the `tReset` value, the `tReset` value should be subtracted from each following `tBase+time`. (Imagine plotting the data point by point along a time x-axis that starts at the event time. The points are plotted ever more to the right, but if that `tReset` threshold is reached, the remaining points are plotted starting back nearer the y-axis. Note that there is an implicit assumption that the event can occur at most once during the sampling that fills the reply buffer.)